**Memory-Efficient-Voxel-Processing-Library**

Introduction:

Files can be so large that it is impractical to load all of their content into memory at once. This problem is often a computational bottleneck in large computed-imaging problems . Matrices that contain mostly zero values are called sparse, distinct from matrices where most of the values are non-zero, called dense. It is computationally expensive to represent and work with sparse matrices as though they are dense, and much improvement in performance can be achieved by using representations and operations that specifically handle the matrix sparsity.

Goal:

To create an API for processing and handling large Sparse Matrices There are many efficient ways to store and work with sparse matrices and our approach is based on split and merge technique

Approach:

Our approach allows to operate large Sparse matrices that would not fit into RAM memory as a whole . The idea is to store the Sparse Matrix in a compressed data structure and retrieve blocks of data from the compressed data structure and dynamically allocating memory and performing the Morphological operations.

Space Complexity:

Very large matrices require a lot of memory, and some very large matrices that we wish to work with are sparse.The matrix contained is sparse with many more zero values than data values. This is clearly a waste of memory resources as those zero values do not contain any information

Time Complexity:

Assuming a very large sparse matrix can be fit into memory, we will want to perform operations on this matrix. Simply, if the matrix contains mostly zero-values, i.e. no data, then performing operations across this matrix may take a long time where the bulk of the computation performed will involve adding or multiplying zero values together. This is a problem of increased time complexity of matrix operations that increases with the size of the matrix.